Operating Systems Lab

Week-2

Exercise 1:

2 children with PID and PPID

CODE:

#include<stdio.h>

#include<unistd.h>

#include<stdlib.h>

#include<sys/wait.h>

#include<sys/types.h>

void child1()

{

printf("\n I'm a child1 ");

printf("\n Child 1 PID is: %d",getpid());

printf("\n Child 1 PPID is: %d",getppid());

}

void child2()

{

printf("\n I'm a child2 ");

printf("\n Child 2 PID is: %d",getpid());

printf("\n Child 2 PPID is: %d",getppid());

}

int main()

{

pid\_t pid1,pid2;

pid1=fork();

if(pid1!=0)

pid2=fork();

if (pid1<0 || pid2<0)

{

printf("\nError ");

}

else if(pid1==0)

{

child1();

}

else if(pid2==0)

{

child2();

}

else

{

printf("\n Parent process ");

printf("\n My pid is: %d",getpid());

printf("\n My ppid is: %d",getppid());

wait(NULL);

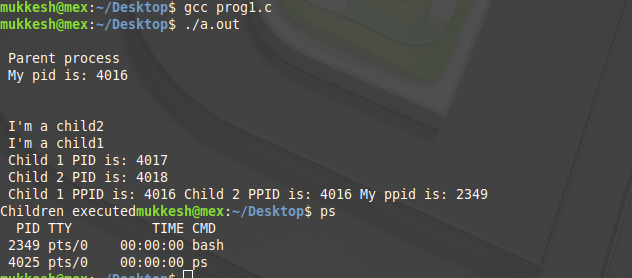
printf("\nChildren executed");

exit(0);

}

}

OUTPUT:



Exercise 2:

2 Children. One should execute ls, while the other should execute cat.

CODE:

#include<stdio.h>

#include<unistd.h>

#include<stdlib.h>

#include<sys/wait.h>

#include<sys/types.h>

void child1()

{

execlp("/bin/ls","ls",NULL);

}

void child2()

{

execlp(“/bin/cat”,”cat”,”meow”,NULL);

}

int main()

{

pid\_t pid1,pid2;

pid1=fork();

if(pid1!=0)

pid2=fork();

if (pid1<0 || pid2<0)

{

printf("\nError ");

}

else if(pid1==0)

{

child1();

}

else if(pid2==0)

{

child2();

}

else

{

printf("\n Parent process ");

wait(NULL);

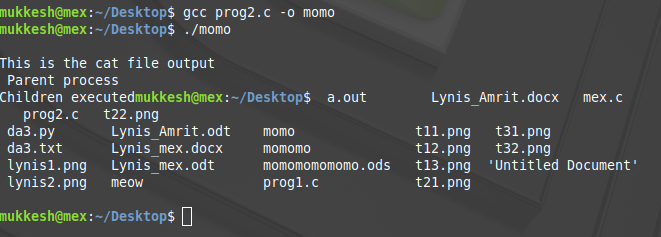
printf("\nChildren executed");

exit(0);

}

}

OUTPUT:



Exercise 3:

2 Children. Child 2 should have child 21 which will execute exercise 2’s output file.

CODE:

#include<stdio.h>

#include<unistd.h>

#include<stdlib.h>

#include<sys/wait.h>

#include<sys/types.h>

void child21()

{

execlp("./momo",NULL);

}

void child1()

{

execlp("/bin/ls","ls",NULL);

}

void child2(int x)

{

int ppp;

if(x==0)

ppp=fork();

if(ppp==0)

{

child21();

}

}

int main()

{

pid\_t pid1,pid2;

pid1=fork();

if(pid1!=0)

pid2=fork();

if (pid1<0 || pid2<0)

{

printf("\nError ");

}

else if(pid1==0)

{

child1();

}

else if(pid2==0)

{

child2(pid2);

}

else

{

printf("\n Parent process ");

wait(NULL);

printf("\nChildren executed");

exit(0);

}

}

OUTPUT:

